TUGAS AKHIR – KI141502

IMPLEMENTASI ALGORITMA *ANT-TREE-MINER* UNTUK KLASIFIKASI IKAN TUNA

REINALDY JALU NUSANTARA
NRP 5110100116

Dosen Pembimbing I
Ahmad Saikhu, S.Si., MT.

Dosen Pembimbing II
Wijayanti Nurul Khotimah, S.Kom., M.Sc.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2015

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESES – KI141502

ANT-TREE-MINER ALGORITHM IMPLEMENTATION FOR CLASSIFYING TUNA FISH

REINALDY JALU NUSANTARA
NRP 5110100116

First Advisor
Ahmad Saikhu, S.Si., MT.

Second Advisor
Wijayanti Nurul Khotimah, S.Kom., M.Sc.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2015

(Halaman ini sengaja dikosongkan)

KATA PENGANTAR

Segala puji bagi Tuhan YME, yang telah melimpahkan karunia-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul *"*IMPLEMENTASI ALGORITMA ANT-TREE-MINER UNTUK KLASIFIKASI IKAN TUNA*"*.

Selesainya tugas akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Sehingga pada kesempatan ini penulis mengucapkan terima kasih kepada:

1. Orang tua, yang telah memberikan segalanya hingga penulis termotivasi untuk menyelesaikan tugas akhir ini.
2. Bapak Ahmad Saikhu, S.Si., MT, selaku pembimbing I yang telah memberikan bimbingan dan bantuan kepada penulis dalam mengerjakan tugas akhir ini.
3. Ibu Wijayanti Nurul Khotimah, S.Kom., M.Sc., selaku pembimbing II yang telah memberikan bimbingan, masukan, motivasi, dan nasehat kepada penulis dalam menyelesaikan Tugas Akhir ini dengan sabar dan penuh perhatian.
4. Hani Ramadhan yang telah menyempatkan waktunya untuk berdiskusi sehingga Tugas Akhir ini dapat diselesaikan.
5. Segenap dosen dan karyawan jurusan Teknik Informatika ITS atas ilmu dan pengalaman yang telah diberikan selama penulis menjalani masa studi di jurusan Teknik Informatika ITS.
6. Teman-teman TC 2010 yang tak pernah berhenti memberikan motivasi, serta bantuan untuk menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan. Sehingga penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depan.

Surabaya, Desember 2015
Reinaldy Jalu Nusantara

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

IMPLEMENTASI ALGORITMA *ANT-TREE-MINER* UNTUK KLASIFIKASI IKAN TUNA

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Cerdas Visual
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh
REINALDY JALU NUSANTARA
NRP : 5110 100 116

Disetujui oleh Dosen Pembimbing Tugas Akhir:

1. Ahmad Saikhu, S.Si., M.T. .....
   NIP: 197107182006041001 (Pembimbing 1)

2. Wijayanti Nurul Khotimah, S.Kom., M.Sc. .....
   NIP: 198603122012122004 (Pembimbing 2)

SURABAYA
DESEMBER, 2015

IMPLEMENTASI ALGORITMA *ANT-TREE-MINER* UNTUK KLASIFIKASI IKAN TUNA

Nama	: Reinaldy Jalu Nusantara
NRP	: 5110100116
Jurusan	: Teknik Informatika – FTIf ITS
Dosen Pembimbing I	: Ahmad Saikhu, S.Si., MT.
Dosen Pembimbing II	: Wijayanti Nurul Khotimah, S.Kom., M.Sc.

Abstrak

Klasifikasi merupakan salah satu bidang dalam *data mining* yang memiliki banyak variasi cara penyelesaian. Para peneliti telah mengembangkan berbagai macam algoritma untuk menyelesaikan masalah ini. Salah satu algoritma yang dikenal luas untuk menyelesaikan klasifikasi adalah *decision tree*, dengan tipe yang banyak digunakan adalah *C4.5 Decision Tree*. Fernando Otero[1] melalui penelitiannya mengembangkan *decision tree* tipe ini dengan mengombinasikannya dengan algoritma *Ant Colony Optimization* untuk pembangunan struktur tree nya. Algoritma baru ini dinamakan dengan algoritma *Ant-Tree-Miner*.

Pada Tugas Akhir ini algoritma tersebut akan diimplementasikan untuk mengklasifikasikan data ikan tuna berbentuk numerik yang diperoleh dari ekstraksi fitur data citra pada Tugas Akhir M. Akbar Kalbuadi[2].

Dari percobaan yang telah dilakukan didapatkan akurasi tertinggi pembangunan tree adalah 82,21% untuk jumlah semut k = 200, dan pembangunan dilakukan dengan pruning.

Kata Kunci: Klasifikasi, Ikan Tuna, *Ant-Tree-Miner*, *Ant Colony Optimization (ACO)*, *Decision Tree*.

(Halaman ini sengaja dikosongkan)

ANT-TREE-MINER ALGORITHM IMPLEMENTATION FOR CLASSIFYING TUNA FISH

Name	: Reinaldy Jalu Nusantara
NRP	: 5110100116
Department	: Informatics Engineering, FTIf, ITS
First Advisor	: Ahmad Saikhu, S.Si., MT.
Second Advisor	: Wijayanti Nurul Khotimah, S.Kom., M.Sc.

Abstract

Classification is one of data mining topic which have many variation of solution. Researchers have developed various algorithm to solve this problem. One of the algorithms popularly used to do the particular job is Decision Tree with its well-known type called C4.5 Decision Tree. Fernando Otero[1] in his research has developed this type of decision tree, combining it with Ant Colony Optimization algorithm for inducing the tree construction. This novel algorithm called Ant-Tree-Miner algorithm.

In this final project, the particular algorithm will be implemented to classify numeric tuna fish data which is obtained from feature extraction of image data in M. Akbar Kalbuadi's final project[2].

Based on the experiment, the best accuracy obtained is 82,21% for colony size k = 200 and the tree is pruned.

Keyword: Classification, Tuna Fish, Ant-Tree-Miner, Ant Colony Optimization (ACO), Decision Tree.

(Halaman ini sengaja dikosongkan)

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

DAFTAR KODE SUMBER

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

Pada bagian ini dijelaskan mengenai hal-hal permulaan yang berhubungan dengan persiapan implementasi tugas akhir meliputi latar belakang, rumusan masalah, batasan masalah, tujuan, metodologi serta sistematika penulisan.

1.1 Latar Belakang

Akhir-akhir ini sektor perikanan Indonesia sedang sangat digalakkan. Hal ini disebabkan karena Indonesia merupakan negara maritim dengan sumber daya bahari yang sangat besar, ditunjukkan dengan banyaknya hewan laut yang ada di perairan Indonesia. Salah satu sumber daya laut yang banyak dikonsumsi dan diperdagangkan adalah ikan, khususnya ikan tuna. Dari data yang dihimpun PT. Aneka Tuna Indonesia pada tahun 2013, mereka mengolah 6 jenis ikan tuna. Jenis tersebut adalah tuna jenis *skipjack*, *yellowfin*, *bigeye*, *blackfin*, *tonggol* dan *albacore* yang masin-masing memiliki penanganan yang berbeda. Selain penanganan, tentu saja harga masing-masing jenis juga berbeda sehingga dibutuhkan proses klasifikasi ikan tuna tersebut. Saat ini proses masih dilakukan secara manual yang melibatkan pegawai dengan jumlah yang besar. Pengklasifikasian oleh manusia rentan terhadap *human error* yang disebabkan kesehatan pegawai ataupun kondisi manusia yang kadang tidak stabil.

Dari penjelasan di atas tampak bahwa diperlukan sebuah sistem otomatis yang dapat melakukan proses klasifikasi ikan-ikan tersebut. Pada tahun 2014, Pengenalan ikan tuna melalui citra sudah pernah dilakukan oleh Muhammad Akbar Kalbuadi [1] dengan menggunakan algoritma *Decision Tree*. Oleh karena itu dengan data citra yang telah diekstrak ke dalam data numerik dari penelitian tersebut, penelitian ini bertujuan melakukan perubahan pada metode klasifikasi yang digunakan. Metode klasifikasi yang digunakan dalam penelitian ini bernama Algoritma *Ant-Tree-*

Miner yang diperkenalkan oleh Fernando E.B Otero dkk [2]. Algoritma ini merupakan pengembangan dari algoritma *Decision Tree C4.5* yang dimodifikasi dengan penggunaan *Ant Colony Optimization (ACO)* pada pembangunan struktur *tree* nya.

1.2 Rumusan Permasalahan

Rumusan masalah yang diangkat dalam Tugas Akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana penerapan algoritma Ant-Tree-Miner dalam mengklasifikasikan jenis ikan tuna?

1.3 Batasan Masalah

Batasan masalah yang ada pada Tugas Akhir ini antara lain:

1. Sistem perangkat lunak dibangun dengan menggunakan perangkat lunak MATLAB R2013a.
2. Data uji coba menggunakan data PT. Aneka Tuna Indonesia yang telah berbentuk numerik setelah dilakukan ekstraksi fitur dari data asal berupa citra pada penelitian sebelumnya.
3. Algoritma klasifikasi yang digunakan adalah *Ant-Tree-Miner*

1.4 Tujuan

Tujuan dari tugas akhir ini adalah mengembangkan aplikasi klasifikasi ikan tuna dengan menerapkan algoritma *Ant-Tree-Miner*, serta menyelidiki bagaimana performa algoritma ini dalam mengklasifikasikan data ikan tuna.

1.5 Manfaat

Manfaat dari Tugas Akhir ini adalah membantu proses klasifikasi ikan tuna dengan lebih cepat dan akurat. Serta mengetahui performa dari algoritma klasifikasi yang digunakan dalam mengklasifikasikan data ikan tuna.

.

1.6 Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan proposal Tugas Akhir.
   Proposal Tugas Akhir ditulis untuk mengajukan ide atas pengerjaan Tugas Akhir. Proposal Tugas Akhir juga mengandung proyeksi hasil dari ide tugas akhir yang diajukan.

2. Studi literatur
   Pada proses ini dilakukan studi lebih lanjut terhadap konsep-kosep yang terdapat pada jurnal, buku, artikel, dan literatur lain yang menunjang. Studi dilakukan untuk mendalami konsep algoritma Ant-Tree-Miner untuk klasifikasi.

3. Implementasi perangkat lunak
   Implementasi merupakan tahap membangun rancangan sistem yang telah dibuat. Pada tahapan ini merealisasikan apa yang terdapat pada tahapan sebelumnya, sehingga menjadi sebuah sistem yang sesuai dengan apa yang telah direncanakan. Pelaksanaan implementasi dilakukan dengan perangkat lunak Matlab.

4. Pengujian dan evaluasi
   Pada tahapan ini dilakukan uji coba terhadap perangkat lunak yang telah dibuat. Pengujian dan evaluasi akan dilakukan dengan melihat kesesuaian dengan perencanaan. Tahap ini dimaksudkan juga untuk mengevaluasi jalannya sistem, mencari masalah yang mungkin timbul dan mengadakan perbaikan jika terdapat kesalahan.

5. Penyusunan buku Tugas Akhir

Pada tahapan ini dilakukan penyusunan buku yang memuat dokumentasi mengenai proses pembuatan program serta hasil dari implementasi perangkat lunak yang telah dibuat.

1.7 Sistematika Penulisan

Buku Tugas Akhir ini disusun dengan sistematika penulisan sebagai berikut:

1. BAB I. PENDAHULUAN

   Bab yang berisi mengenai latar belakang, tujuan, dan manfaat dari pembuatan Tugas Akhir. Selain itu permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penulisan juga merupakan bagian dari bab ini.

2. BAB II. TINJAUAN PUSTAKA

   Bab ini berisi teori pendukung dan literatur yang berkaitan dengan bahasan dan mendasari pembutan tugas akhir ini. Teori-teori yang dijelaskan antara lain mengenai algoritma *decision tree*, *Ant Colony Optimization*, dan algoritma *Ant-Tree-Miner*.

3. BAB III. PERANCANGAN SISTEM

   Bab ini berisi penjelasan mengenai perancangan data dan desain metode yang akan digunakan dalam tugas akhir.

4. BAB IV. IMPLEMENTASI

   Bab ini berisi penjelasan implementasi desain algoritma dan implementasi dari pembuatan tugas akhir.

5. BAB V. PENGUJIAN DAN EVALUASI

   Bab ini berisi penjelasan mengenai data hasil percobaan dan pembahasan mengenai hasil percobaan yang telah dilakukan.

6. BAB VI. KESIMPULAN DAN SARAN

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan ke depannya.

(Halaman ini sengaja dikosongkan)

BAB II
DASAR TEORI

Bab ini berisi penjelasan teori-teori yang berkaitan dengan algoritma yang diajukan pada pengimplementasian sistem. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap sistem yang dibuat dan berguna sebagai penunjang dalam pengembangan sistem.

2.1 Data Numerik Ikan Tuna

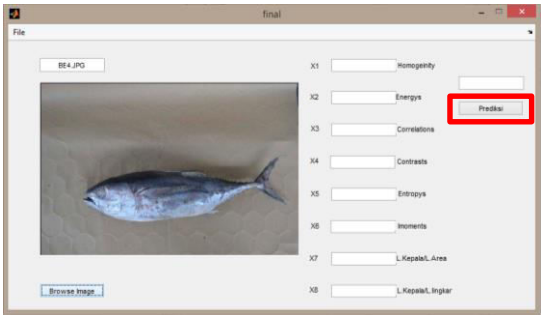
Data ini didapatkan dari hasil penelitian sebelumnya [1]. Yaitu dari data masukan berupa citra yang dimasukkan ke dalam sistem pada penelitian tersebut untuk mendapatkan ekstraksi fitur berupa data numerik. Contoh data citra sesungguhnya dapat dilihat pada Gambar 2.1.

Gambar 2.1 Data Masukan Citra Ikan Tuna

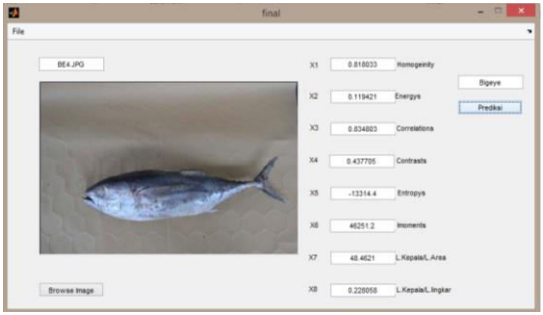
Citra tersebut kemudian dimasukkan ke dalam sistem yang telah dibuat pada penelitian tersebut lalu dilakukan pemrosesan (Gambar 2.2).

Dalam Gambar 2.3 terlihat bahwa setelah ekstraksi fitur terdapat delapan nilai yaitu: x1(*homogenity*), x2(*energys*), x3(*correlations*), x4(*contrasts*), x5(*entropys*), x6(*Imoments*), x7(*L.Kepala/L.Area*), x8(*L.Kepala/L.Lingkar*). Masing-masing

nilai tersebut akan menjadi sebuah atribut dalam proses klasifikasi dalam tugas akhir ini. Proses tersebut dilakukan sebanyak jumlah data yaitu enam puluh data citra. Kemudian setiap ekstraksi dari data citra tersebut menjadi sebuah record dalam file berekstensi .csv yang akan menjadi data masukan untuk tugas akhir ini.



Gambar 2.2 Citra Tuna Dimasukkan dalam Sistem untuk Diproses



Gambar 2.3 Hasil Ekstraksi Fitur

2.2 *C4.5 Decision Tree*

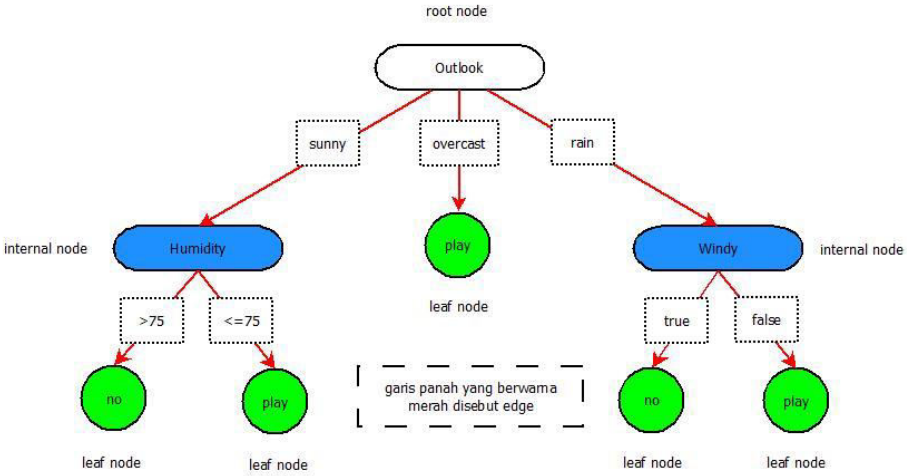
Decision Tree merupakan salah satu metode yang digunakan untuk masalah klasifikasi. Algoritma ini memiliki beberapa unsur yang selalu ada seperti:

a. *Node*

Sebuah titik dalam *Decision Tree*. Ada tiga jenis *node* yaitu:

- *Root Node*
  Adalah *node* yang berada pada paling atas, sehingga merupakan *node* yang paling awal terbentuk
- *Internal Node*
  Adalah *node* yang memiliki *node* lain di atasnya dan merepresentasikan pengujian atribut.
- *Leaf Node*
  Adalah *node* yang merepresentasikan label atribut kelas yang dituju berdasarkan pengujian-pengujian di *internal node* diatasnya.

b. *Edge*
Garis dalam *decision tree* yang menghubungkan antara satu *node* dengan *node* lainnya.



Gambar 2.4 Contoh Decision Tree

Dalam mengklasifikasikan suatu data, sebuah *tree* ditelusuri dari *root node* sampai ke *leaf node*. Proses ini dimulai dari *root node*, lalu berpindah ke *node* satu tingkat dibawahnya yang terhubung dengan *edge*. Apabila satu tingkat dibawah suatu *node* terhubung dengan beberapa *node*, maka *node* bawah yang dituju adalah yang sesuai dengan hasil tes pada *internal node*. Hal ini terus berlanjut sampai *leaf node* tercapai. Contoh *decision tree* dapat dilihat pada Gambar 2.4.

OUTL.	TEMP.	HUMID.	WINDY	PLAY
Sunny	85	85	False	No
Sunny	80	90	True	No
Overcast	83	78	False	Play
Rain	70	96	False	Play
Rain	68	80	False	Play
Rain	65	70	True	No
overcast	64	65	True	Play
Sunny	72	95	False	No
Sunny	69	70	False	Play
Rain	75	80	False	Play
Sunny	75	70	True	Play
Overcast	72	90	True	Play
Overcast	81	75	False	Play
Rain	71	80	True	No

Satu buah record

Memiliki dua label kelas yaitu "Play" dan "No"

Gambar 2.5 Contoh Data

Algoritma yang sangat terkenal untuk membangun *decision tree* adalah algoritma C4.5 yang merupakan pengembangan dari algoritma ID3 serta merupakan jenis *decision tree* yang dipakai sebagai dasar algoritma *Ant-Tree-Miner*. Dalam algoritma C4.5 ini, pembangunan struktur *tree* menggunakan

kriteria berbasis *entropy* dalam memilih atribut terbaik untuk dijadikan *node*, yang disebut *Information Gain Ratio*. Pada Gambar 2.5 diberikan contoh data untuk mendemonstrasikan penghitungan nilai *Information Gain Ratio*.

a. *Entropy*
Suatu nilai yang dipakai untuk menunjukkan ketidakmurnian dari sekumpulan *record* relatif terhadap nilai atribut kelasnya. Nilai *entropy* dari sekumpulan *record* S dapat dihitung dengan Persamaan 2.1

$$Entropy(S) = \sum_{c=1}^{m} - p_c \cdot \log_2 p_c \qquad (2.1)$$

p_c = proporsi *record* dalam S yang memiliki label kelas ke-*c*
m = Jumlah label kelas yang ada dalam S

Pada contoh di atas, p_1 adalah perbandingan jumlah *record* yang memiliki label kelas "Play" yaitu lima *record* dengan jumlah keseluruhan *record* yang berjumlah empat belas. Kemudian p_2 adalah perbandingan jumlah *record* yang memiliki label kelas "No" yaitu sembilan *record* dengan jumlah keseluruhan *record* yang berjumlah empat belas.

$$Entropy(S) = \left(-\frac{5}{14} \times \log_2 \frac{5}{14} \right) + \left(-\frac{9}{14} \times \log_2 \frac{9}{14} \right) = 0{,}94$$

b. *Information Gain*
Information Gain dari sebuah atribut A adalah reduksi yang diharapkan pada *entropy* dengan cara membagi *data training*

ke dalam T *subset*, dimana T adalah berapa banyak nilai unik yang ada dalam atribut A. Information Gain dihitung menggunakan Persamaan 2.2.

$$Gain(S, A) = Entropy(S) - \sum_{v=1}^{T} \frac{|S_v|}{|S|} \cdot Entropy(S_v) \qquad (2.2)$$

$|S_v|$ = jumlah *record* dalam subset S yang atribut A-nya memiliki nilai ke-v
$|S|$ = jumlah *record* dalam S

Misalkan pada kasus *Gain(S,Outlook)*, pada kasus tersebut $|S_1|$ adalah banyaknya *record* yang bernilai "Sunny" yaitu lima buah. Kemudian p_1 dalam *Entropy*(S_1) adalah jumlah *record* yang berlabel "Yes" dalam kumpulan record yang atribut Outlook-nya bernilai "Sunny". Begitu seterusnya sampai $v = 3$ karena nilai atribut Outlook ada tiga jenis yaitu *Sunny*, *Overcast*, dan *Rain*.

$$\frac{|S_1|}{|S|} \cdot Entropy(S_1) = \frac{5}{14} \cdot \left[\left(-\frac{2}{5} \cdot \log_2 \frac{2}{5} \right) + \left(-\frac{3}{5} \cdot \log_2 \frac{3}{5} \right) \right] = 0{,}35$$

$$\frac{|S_2|}{|S|} \cdot Entropy(S_1) = \frac{4}{14} \cdot \left[\left(-\frac{4}{4} \cdot \log_2 \frac{4}{4} \right) + 0 \right] = 0$$

$$\frac{|S_3|}{|S|} \cdot Entropy(S_3) = \frac{5}{14} \cdot \left[\left(-\frac{3}{5} \cdot \log_2 \frac{3}{5} \right) + \left(-\frac{2}{5} \cdot \log_2 \frac{2}{5} \right) \right] = 0{,}35$$

$$\sum_{v=1}^{3} S_v = 0{,}35 + 0 + 0{,}35 = 0{,}7$$

$$Gain(S, Outlook) = 0{,}94 - 0{,}7 = 0{,}24$$

c. *Split Information*

Split Information adalah suatu pengurangan nilai untuk atribut-atribut yang membagi *data training* ke *subset* yang sangat kecil. *Split Information* dihitung dengan Persamaan 2.3.

$$Split \quad Information(S, A) = \sum_{v=1}^{T} -\frac{|S_v|}{|S|} \cdot \log_2 \frac{|S_v|}{|S|} \qquad (2.3)$$

d. *Information Gain Ratio*

Nilai yang didapatkan dari pembagian nilai *Gain* dan *Split Information*. Nilai ini yang nantinya digunakan sebagai informasi heuristik pada algoritma *Ant-Tree-Miner*. Information Gain Ratio dihitung dengan Persamaan 2.4:

$$Gain \quad Ratio(S, A) = \frac{Gain(S, A)}{Split \quad Information(S, A)} \qquad (2.4)$$

2.3 *Ant Colony Optimization (ACO)*

Ant-Colony-Optimization merupakan sebuah teknik untuk memecahkan problem-problem yang dapat diterjemahkan sebagai pencarian jalur terbaik melalui graf. Teknik ini terinspirasi dari perilaku koloni semut dalam mencari jalur dari sumber makanan menuju ke sarangnya. Semut-semut ini menggunakan zat kimia (feromon) sebagai cara untuk berkomunikasi. Semut-semut

meninggalkan feromon saat mereka berjalan dari sumber makanan ke sarang, yang menyebabkan terbentuknya jejak feromon dari jalur yang dipakai. Konsentrasi feromon di jalan mempengaruhi pilihan yang diambil semut lain. Semakin banyak feromon yang terdapat di jalan akan memicu semut lain untuk menggunakan jalan tersebut. Karena jalur yang lebih pendek dilalui dengan lebih cepat, maka jalur tersebut memiliki konsentrasi feromon yang lebih kuat. Jalur tersebut akan lebih dipilih oleh semut lain, sehingga lama kelamaan semua semut akan memakai jalur yang sama.

2.3.1 Perilaku Semut

Seekor semut *k* pada *node i* akan memilih *node j* yang dituju pada layer berikutnya dengan probabilitas pada Persamaan 2.5.

$$p_{i,j} = \begin{cases} \dfrac{\tau_{i,j}^{\alpha}}{\sum_{j \in N_i^{(k)}} \tau_{i,j}^{\alpha}} & \text{, jika } j \in N_i^{(k)} \\ 0 & \text{, jika j} \notin N_i^{(k)} \end{cases} \qquad (2.5)$$

dimana α menunjukkan derajat kepentingan pheromone dan $N_i^{(k)}$ adalah pilihan yang dipunyai semut *k* (neighborhood) pada saat ia berada pada *node i*. Neighborhood dari semut *k* pada simpul *i* akan mengandung semua *node* yang bisa dituju yang tersambung secara langsung ke *node i*, kecuali *node* yang sudah dikunjungi sebelumnya.

2.3.2 Penambahan dan Penguapan Pheromone

Seekor semut k ketika melewati ruas akan meninggalkan pheromone. Jumlah pheromone yang terdapat pada ruas *ij* setelah dilewati semut *k* diberikan dengan rumus pada Persamaan 2.6.

$$\tau_{i,j} \leftarrow \tau_{i,j} + \Delta\tau_{i,j}^{k} \qquad (2.6)$$

Dengan meningkatnya nilai pheromone pada ruas *i–j*, maka kemungkinan ruas ini akan dipilih lagi pada iterasi berikutnya semakin besar. Setelah sejumlah simpul dilewati maka akan terjadi penguapan pheromone dengan aturan pada Persamaan 2.7.

$$\tau_{i,j} \leftarrow (1 - \rho)\tau_{i,j}, \forall (i,j) \in A \qquad (2.7)$$

dimana $\rho \in (0,1]$ adalah parameter tingkat penguapan dan A menyatakan segmen atau ruas yang sudah dilalui oleh semut k sebagai bagian dari lintasan dari sarangnya menuju ke makanan. Penurunan jumlah pheromone memungkinkan semut untuk mengeksplorasi lintasan yang berbeda selama proses pencarian. Ini juga akan menghilangkan kemungkinan memilih lintasan yang kurang bagus. Selain itu, ini juga membantu membatasi nilai maksimum yang dicapai oleh suatu lintasan pheromone. Jumlah pheromone yang ditambahkan pada ruas $i - j$ oleh semut k diberikan dengan Persamaan 2.8.

$$\Delta\tau_{i,j}^{k} = \begin{cases} \frac{cf_{best}}{f_{worst}} & \text{, jika } j \in N_i^{(k)} \\ 0 & \text{, jika j} \notin N_i^{(k)} \end{cases} \qquad (2.8)$$

Dimana f_{best} adalah nilai terbaik dari fungsi tujuan dan f_{worst} nilai terjelek dari fungsi tujuan. Sedangkan c adalah konstanta untuk mengontrol skala updating global pheromone. Semakin tinggi nilai c semakin banyak pheromone ditambahkan ke lintasan global terbaik dan semakin bagus kemampuan mengeksploitasi.

2.4 Algoritma Ant-Tree-Miner (ATM)

ATM Merupakan algoritma yang pertama kali diperkenalkan oleh Fernando E.B Otero, dkk [2] yang merupakan pengembangan dari *Decision Tree* tipe C4.5 dengan menggunakan *Ant-Colony-Optimization* dalam pembangunan struktur *tree*-nya. Dalam metode C4.5 pemilihan atribut yang akan dipakai sebagai node hanya didasarkan pada informasi heuristik atau nilai

information gain ratio yang terbesar, sedangkan pada algoritma Ant-Tree-Miner pemilihan atribut dilakukan secara probabilistik dengan mempertimbangkan informasi heuristik dan jumlah feromon semut. Diagram alir algoritma Ant-Tree-Miner dapat dilihat pada Gambar 2.6.

2.4.1 Inisialisasi Feromon

Menentukan jumlah feromon awal yang sama untuk semua edge yang ada. Dalam tugas akhir ini digunakan jumlah feromon awal adalah 1.

2.4.2 Hitung Informasi Heuristik

Penghitungan informasi heuristik menggunakan persamaan berikut:

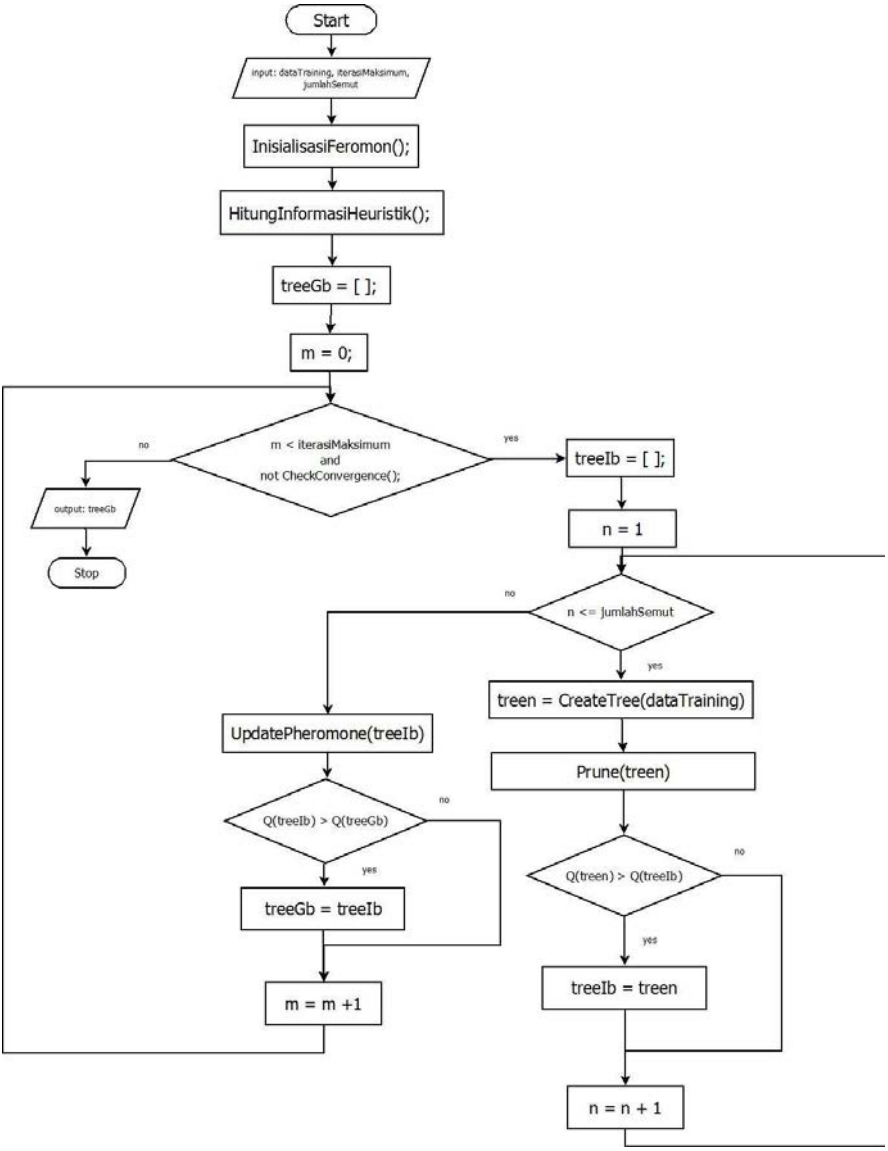
$$\eta_{x_i} = Gain\ Ratio(S, x_i) \tag{2.9}$$

S = set data training
x_i = vertex atribut ke-i

Penghitungan nilai *Gain Ratio* sama seperti proses pada persamaan (2.4). Penghitungan *Gain ratio* didapatkan dengan terlebih dahulu menghitung nilai *entropy* (persamaan 2.1), dilanjutkan penghitungan nilai *information gain* (persamaan 2.2), serta terakhir penghitungan nilai *split information* (persamaan 2.3).

2.4.3 Buat *Tree*

Dalam membangun tree terdapat perbedaan dalam penanganan data nominal dan dan kontinyu. Dikarenakan semua atribut pada data ikan tuna bertipe numerik maka harus melalui proses diskritisasi terlebih dahulu. Gambar 2.7(a dan b) adalah *pseudocode* dari proses *BuatTree.*



Gambar 2.6 Diagram Alir Algoritma Ant-Tree-Miner

Input: data *training (Set Data)*, daftar atribut prediktor (*Set Atribut*), *edge* saat ini (*Edge*)
Output: *root node* dari *decision tree* yang dibuat

1. *A* ← memilih suatu atribut dalam *Set Atribut* secara probabilistik untuk dikunjungi dari *Edge*;
2. *root* ← membuat *decision node* baru yang merepresentasikan atribut *A*
3. *Set Kondisi* ← Ø
4. **if** *A* adalah atribut nominal **then**
5. *Set Atribut* ← *Set Atribut* – {*A*};
6. **for all** v_i dalam A **do**
7. *Set Kondisi* ← *Set Kondisi* + {$A = v_i$ }
8. **end for**
9. **else**
10. Set Kondisi ← diskritisasi(Set Atribut, Set Data);
10. **end if**
11. **for all** kondisi atribut *T* dalam *Set Kondisi* **do**
12. $cabang_i$ ← cabang baru yang merepresentasikan *T* dari *root*
13. $subset_i$ ← subset dari *Set Data* yang memenuhi kondisi *T*
14. **if** $subset_i$ kosong **then**
15. Buat *leaf node* dengan label kelas mayoritas dari *Set Data* dibawah $cabang_i$;
16. **else if** semua data dalam $subset_i$ mempunyai label kelas yang sama **then**
17. Buat *leaf node* dengan label kelas dari $subset_i$ dibawah $cabang_i$
18. **else if** jumlah data dalam $subset_i$ kurang dari *threshold* **then**
19. Buat *leaf node* dengan label kelas mayoritas dari $subset_i$ dibawah $cabang_i$
20. **else if** *Set Atribut* kosong **then**

Gambar 2.7(a) *Pseudocode* Fungsi BuatTree

```
21.             Buat leaf node dengan label kelas mayoritas dari
                subset_i dibawah cabang_i
22.     else
23.             Buat subtree dengan nilai balik dari fungsi
                BuatTree(subset_i, Set Atribut, cabang_i) dibawah
                cabang_i
24.     end if
25. end for
26. return root;
```

Gambar 2.7(b) Pseudocode Fungsi Buat Tree (b)

2.4.3.1 Proses Diskritisasi

Proses pendiskritisasian dilakukan seperti yang dilakukan pada algoritma C4.5, yaitu memilih nilai *threshold* yang menghasilkan nilai *gain ratio* tertinggi. Nilai *threshold* ini akan membagi set menjadi dua subset. Subset pertama yaitu *record* yang nilainya kurang dari sama dengan nilai *threshold*. Subset kedua yaitu *record* yang nilainya lebih besar dari nilai *threshold*.

Berikut ini adalah ilustrasi proses diskritisasinya. Pertama urutkan nilai-nilai atribut kontinyu dengan urut naik. Kandidat nilai ambang batas adalah nilai yang muncul berurutan yang memiliki kelas klasifikasi yang berbeda.

Misalkan dari 48 data training yang ada, untuk atribut "x1" setelah diurutkan urut naik, sepuluh data pertama diambil sebagai contoh seperti pada tabel 2.3.2.1. Kandidat nilai ambang batas adalah nilai-nilai berwarna kuning yaitu {0.756512, 0.757013, 0.760246}. Nilai ambang batas membagi set suatu atribut menjadi dua subset. Yaitu yang tidak lebih besar dari nilai ambang batas dan lebih besar dari nilai ambang batas. Setelah itu diuji untuk setiap nilai kandidat, manakah kandidat nilai yang menghasilkan nilai *gain ratio* yang paling tinggi. Pada tabel 2.1 diberikan contoh pemilihan kandidat nilai *threshold*. Baris yang berwarna kuning merupakan nilai-nilai yang menjadi kandidat *threshold*.

Tabel 2.1 Pemilihan Kandidat Threshold

No	x1	class
1	0.738843	Skipjack
2	0.739117	Skipjack
3	0.743579	Skipjack
4	0.748133	Skipjack
5	0.751931	Skipjack
6	0.756512	Skipjack
7	0.757013	Yellowfin
8	0.757605	Skipjack
9	0.760246	Skipjack
10	0.76877	Yellowfin

a. Untuk kandidat nilai ambang batas pertama (0.756512):

$$Entropy(S) = \sum_{c=1}^{m} - p_c \cdot \log_2 p_c$$

$$Entropy(S) = \left(- p_1 \cdot \log_2 p_1\right) + \left(- p_2 \cdot \log_2 p_2\right)$$

$$Entropy(S) = \left(-\frac{8}{10} \cdot \log_2 \frac{8}{10} \right) + \left(-\frac{2}{10} \cdot \log_2 \frac{2}{10} \right) = 0{,}722$$

$$Gain(S, A) = Entropy(S) - \sum_{v=1}^{T} \frac{|S_v|}{|S|} \cdot Entropy(S_v)$$

$$\frac{|S_{\leq 0{,}756512}|}{|S|} \cdot Entropy(S_{\leq 0{,}756512}) = \frac{6}{10} \cdot \left[\left(-\frac{6}{6} \cdot \log_2 \frac{6}{6} \right) + 0 \right] = 0$$

$$\frac{|S_{>0,756512}|}{|S|} \cdot Entropy(S_{>0,756512}) = \frac{4}{10} \cdot \left[\left(-\frac{2}{4} \cdot \log_2 \frac{2}{4} \right) + \left(-\frac{2}{4} \cdot \log_2 \frac{2}{4} \right) \right]$$

$$= 0,4$$

$$Gain = 0,722 - (0 + 0,4) = 0,322$$

$$Split \quad Information(S, A) = \sum_{v=1}^{T} -\frac{|S_v|}{|S|} \cdot \log_2 \frac{|S_v|}{|S|}$$

$$Split \quad Information = \left(-\frac{|S_{\leq 0,756512}|}{|S|} \cdot \log_2 \frac{|S_{\leq 0,756512}|}{|S|} \right) +$$

$$\left(-\frac{|S_{>0,756512}|}{|S|} \cdot \log_2 \frac{|S_{>0,756512}|}{|S|} \right)$$

$$= \left(-\frac{6}{10} \cdot \log_2 \frac{6}{10} \right) + \left(-\frac{4}{10} \cdot \log_2 \frac{4}{10} \right)$$

$$= 0,971$$

$$Gain \quad Ratio(S, A) = \frac{Gain(S, A)}{Split \quad Information(S, A)}$$

$$= \frac{0,322}{0,971}$$

$$= 0,332$$

b. Dengan cara yang sama untuk nilai *threshold* kedua (0.757013) didapatkan nilai *Gain Ratio* 0.037
c. Dengan cara yang sama untuk nilai ambang batas ketiga (0.760246) didapatkan nilai *Gain Ratio* 0.574

Sehingga nilai threshold yang dipilih adalah 0.760246 karena menghasilkan *Gain Ratio* tertinggi yaitu 0.574.

2.4.3.2 Pemilihan Atribut oleh Semut

Pembangunan kandidat *decision tree* dilakukan oleh setiap semut. Pada *decision tree* C4.5, pemilihan atribut dilakukan secara deterministic dengan hanya melibatkan nilai informasi huristik, akan tetapi pada algoritma *Ant-Tree-Miner* dilakukan secara stokastik berdasarkan informasi heuristik dan nilai feromon. Pada setiap iterasi dalam proses pembangunan *tree*, seekor semut menerapkan aturan probabilistik untuk menentukan *vertex* mana yang akan dikunjungi berdasarkan jumlah feromon dan informasi heuristik. Probabilitas p_i dari seekor semut untuk mengunjungi atribut vertex x_i dihitung dengan Persamaan 2.10.

$$p_i = \frac{\tau_{(E,L,x_i)} \cdot \eta_i}{\sum_{i \in F} \tau_{(E,L,x_i)} \cdot \eta_i} \quad , \forall i \in F \tag{2.10}$$

- $\tau_{(E, L, xi)}$ adalah jumlah feromon yang bersesuaian dengan masukan (E, L, x_i). E adalah kondisi atribut yang direpresentasikan oleh *edge* yang dilalui atau '−' pada awal mula proses pembangunan. L adalah level semut saat ini pada *decision tree* atau 0 saat awal mula proses pembangunan. x_i adalah vertex atribut ke-i dalam graf yang dibangun.
- η_i adalah informasi heuristic dari atribut ke-i.

- *F* adalah set dari atribut yang bisa dipilih.

2.4.4 *Prune* (Pemangkasan) *Tree*

Setelah kandidat *decision tree* dibuat maka dilakukan proses pemangkasan. Tujuan dari proses pemangkasan adalah untuk memperbaiki kemampuan generalisasi dan tentunya akurasi prediksi dengan cara menghapus *decision node* yang tidak dibutuhkan. Proses *pruning* dalam tugas akhir ini dilakukan seperti dalam dengan metode yang disebut *confidence-interval-based-pruning* [3]. Pruning ini didasarkan pada perhitungan pada Persamaan 2.11.

$$p = f + z\sqrt{\frac{f \cdot (1 - f)}{N}} \tag{2.11}$$

p = nilai error
f = rasio klasifikasi yang salah dengan jumlah keseluruhan klasifikasi oleh suatu leaf
z = Faktor z (dalam tugas akhir ini dipakai z = 0.69)
N = jumlah keseluruhan klasifikasi oleh suatu leaf

Apabila nilai p suatu parent lebih kecil dari nilai p rata-rata semua childnya maka dilakukan pruning.

2.4.5 Hitung Kualitas *Tree*

Penghitungan kualitas sebuah tree dilakukan setelah proses pruning. Kualitas sebuah tree dihitung dengan Persamaan 2.12.

$$Q = \frac{N - Error}{N} \tag{2.12}$$

Q = Kualitas *tree*
N = jumlah semua *record* yang diklasifikasi

Error = Jumlah *record* yang salah diprediksi

2.4.6 *Update* Feromon

Proses *update* feromon dilakukan di setiap iterasi setelah $tree_{ib}$ didapatkan. Feromon selalu berkurang dikarenakan tingkat penguapan (*evaporation rate*) ρ yang merupakan parameter yang ditentukan oleh pengguna. Khusus untuk feromon pada *tree* terbaik lokal, feromon akan ditambah dengan nilai kualitas *tree*. Proses tersebut diberikan dalam Persamaan 2.13.

$$\tau_{(E,L,x_i)} = \begin{cases} \rho \cdot \tau_{(E,L,x_i)} & \text{, Jika } (E, L, x_i) \notin tree_{ib} \\ \rho \cdot \tau_{(E,L,x_i)} + Q(tree_{ib}) & \text{, Jika } (E, L, x_i) \in tree_{ib} \end{cases}$$

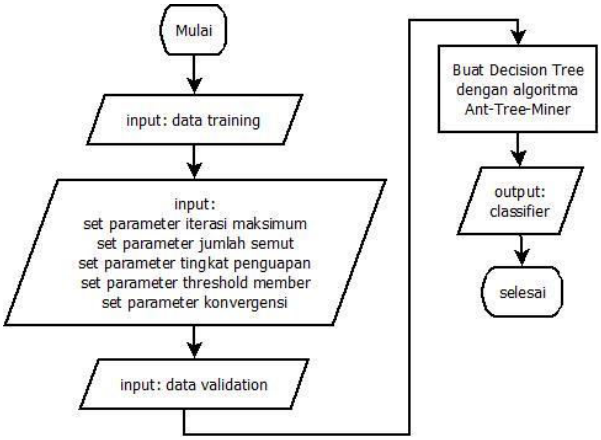
(2.13)

BAB III
PERANCANGAN PERANGKAT LUNAK

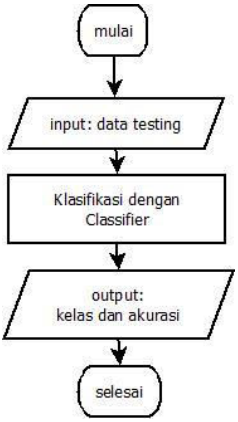
Pada bab ini akan dijelaskan mengenai hal-hal yang berkaitan dengan perancangan sistem yang akan dibuat. Perancangan sistem pada bagian ini meliputi tiga bagian penting yaitu penjelasan data masukan, data keluaran, dan algoritma yang digunakan dalam sistem.

3.1 Deskripsi Umum

Pada Tugas Akhir ini dirancang sebuah aplikasi berbasis matlab untuk mengklasifikasikan data ikan tuna. Sistem melakukan dua proses utama yaitu training dan testing. Proses training ditunjukkan pada Gambar 3.1 sedangkan proses testing ditunjukkan pada Gambar 3.2.



Gambar 3.1 Diagram Alir Proses Training



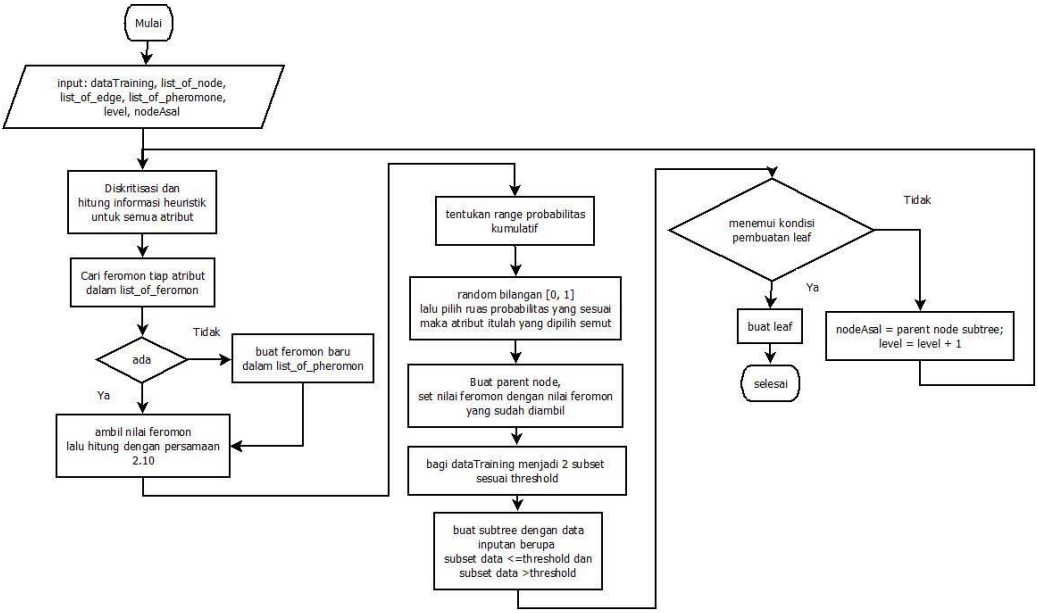
Gambar 3.2 Diagram Alir Proses Testing

3.2 Pembuatan Decision Tree dengan Algoritma Ant-Tree-Miner

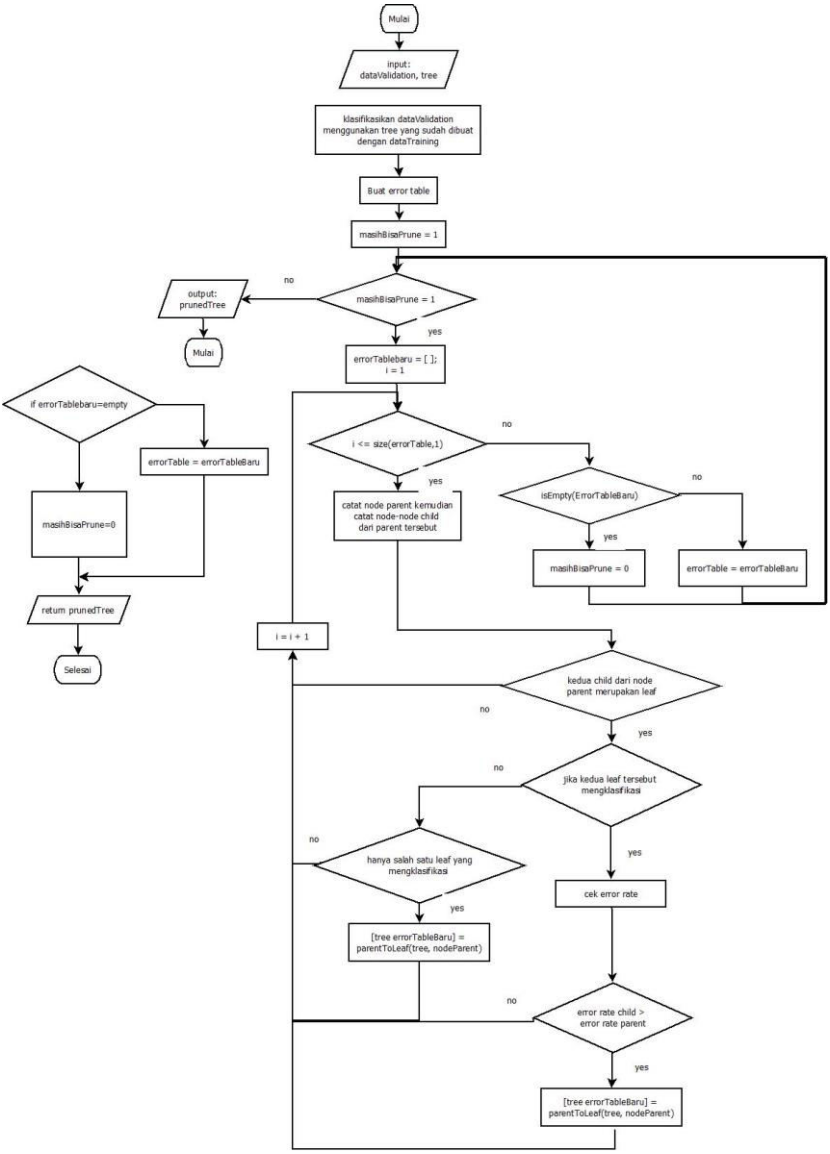
Pembuatan decision tree mencakup semua proses yang sudah dijelaskan oleh diagram alir pada Gambar 2.6. Terdapat dua proses penting dalam pembuatan decision tree yaitu pembangunan tree dan proses pruning.

3.2.1 Pembangunan Tree

Proses ini merupakan proses yang sangat vital karena banyak data yang mendukung proses-proses lain dibuat dalam proses pembangunan tree. Gambar 3.3 adalah diagram alir proses pembangunan tree.



Gambar 3.3 Diagram Alir Proses Pembangunan Tree



Gambar 3.4 Diagram Alir Proses Pruning Tree

Pada proses pembangunan tree ini pertama-tama sistem akan melakukan diskritisasi sekaligus menghitung nilai informasi heuristik untuk semua atribut. Proses diskritisasi dilakukan seperti penjelasan bab 2.4.3.1, sedangkan penghitungan nilai informasi heuristik melalui persamaan 2.9. Pencarian feromon pada list_of_pheromone dilakukan karena ada kemungkinan antara satu semut dengan semut yang lain membuat lintasan tree yang sama. Pada diagram alir di atas, kondisi pembuatan leaf adalah seperti yang telah dijelaskan pada pseudocode Gambar 2.7 (a dan b) pada bab 2.4.3 yaitu ketika subset data kosong, semua data dalam subset data memiliki label kelas yang sama, serta jumlah record dalam subset data kurang dari threshold member.

3.2.2 Proses Pruning Tree

Proses pruning menggunakan bantuan tabel error untuk menghitung *error-rate* sebuah tree. Error table terdiri dari tiga kolom yaitu indeks *leaf* yang mengklasifikasi, jumlah klasifikasi salah oleh *leaf* tersebut, dan jumlah record total yang diklasifikasi oleh *record* tersebut. Pruning dilakukan apabila kedua *child* dari suatu *parent* mengklasifikasi data validation dan juga penghitungan *error rate* menunjukkan harus dilakukan pruning. Selain itu pruning juga dilakukan bila hanya salah satu *leaf* yang mengklasifikasi data validation. Proses pruning dijelaskan pada diagram alir Gambar 3.4.

3.3 Perancangan Data

Dalam Tugas Akhir ini, pada perancangan data dibagi menjadi dua macam data yaitu data masukan, dan data keluaran

3.3.1 Data Masukan

Data masukan dari aplikasi ini merupakan data numerik yang dihasilkan dari ekstraksi fitur data asal berupa citra. Data yang telah diekstrak tersebut memiliki delapan atribut prediktor yaitu:

homogenity, *energy*, *correlation*, *contrasts*, *entropy*, *inverse moment*, perbandingan luas kepala ikan dengan persegi dengan sisi panjang dari kepala ikan, dan perbandingan luas kepala ikan dengan setengah lingkaran yang memiliki jari-jari sebesar panjang kepala ikan tersebut. Data ini berbentuk file berekstensi *.csv* dengan jumlah sebanyak 60 buah *record*. Data ini dibagi menjadi tiga, data *training*, data *validation*, dan data *testing*. Jumlah *record* untuk masing-masing data akan dijelaskan pada Bab 5.

Data *training* digunakan untuk membangun struktur model *decision tree*, data *validation* digunakan untuk memperbaiki sruktur *decision tree* yang sudah terbentuk, sedangkan data *testing* digunakan untuk mengukur sejauh mana model yang telah dibuat berhasil melakukan klasifikasi dengan benar.

Diibaratkan data berbentuk tabel, ketiga data ini berukuran (jumlah atribut+1) dikalikan jumlah record masing-masing data. Pada ketiga jenis data ini selain terdapat delapan kolom yang merepresentasikan jumlah atribut prediktor, juga terdapat satu kolom tambahan untuk menampung nilai kelas sebenarnya. Nilai kelas sebenarnya berfungsi untuk menghitung akurasi klasifikasi.

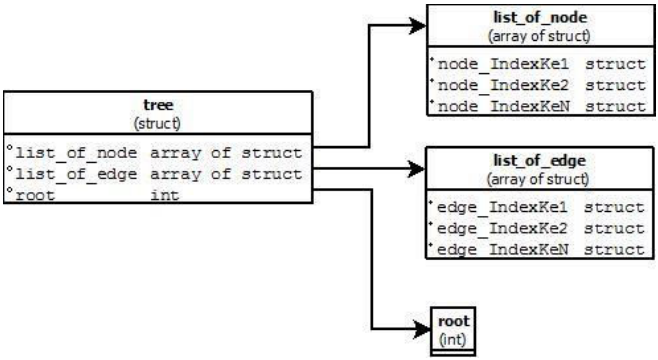
Selain itu lima parameter yang ditentukan oleh user juga dimasukkan yaitu jumlah semut, iterasi maksimum, koefisien penguapan feromon, *threshold* jumlah anggota minimal (*pseudocode* Gambar 2.7 baris 18), serta jumlah minimal akurasi yang sama untuk dianggap algoritma sudah konvergen.

3.3.2 Data Luaran

Data luaran yang dihasilkan oleh sistem ini adalah hasil kelas klasifikasi dari tiap record data uji yang diprediksi dengan algoritma Ant-Tree-Miner. Kelas klasifikasi terdiri dari 3 kelas yaitu “1” untuk kelas “Big Eye”, “2” untuk kelas “Skipjack”, dan “3” untuk kelas “Yellowfin”. Hasil kelas klasifikasi ini akan dibandingkan dengan kelas asli dari data testing sehingga melalui perbandingan hasil klasifikasi benar dan hasil klasifikasi salah akan didapatkan nilai persentase akurasi dari algoritma ini dalam satu kali uji coba.

3.4 Representasi Struktur Data *Tree*

Dalam implementasinya, *Decision tree* direpresentasikan dengan struktur data ***struct*** dengan tiga buah member. Yaitu "list_of_node", "list_of_edge", dan "root". Ilustrasi dari struktur data tersebut dapat dilihat pada Gambar 3.5. Penjelasan tiap member dari struct "tree" terdapat dalam Tabel 3.1.



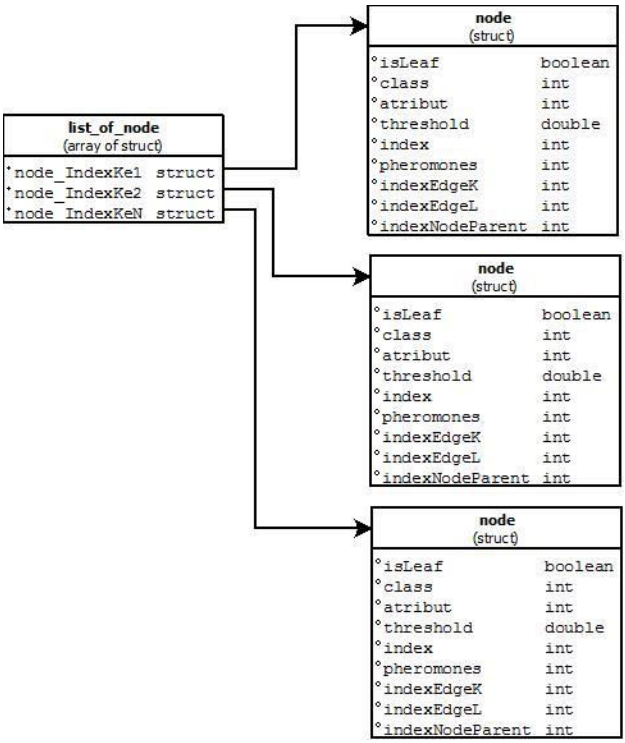
Gambar 3.5 Representasi Struktur Data Tree

Tabel 3.1 Penjelasan Member Struct "tree"

No	Anggota	Tipe	Penjelasan
1.	list_of_node	Array of struct	Berisi kumpulan *node* yang ada dalam *tree*. *Node* diakses melalui indeks *array* dimana *edge* tersebut disimpan.
2.	list_of_edge	Array of struct	Berisi kumpulan *edge* yang ada dalam *tree*. *Edge* diakses melalui indeks *array* dimana *edge* tersebut disimpan.
3.	root	Int	Merupakan indeks array dimana root node sebuah tree/subtree disimpan dalam "list_of_node"

3.4.1 Representasi Struktur Data *Node*

Dalam Tugas Akhir ini sebuah node direpresentasikan sebagai sebuah ***struct*** dengan lima buah member yaitu “isLeaf”, “class”, “atribut”, “threshold”, “index”, “pheromones”, “indexEdgeK”, “indexEdgeL”, “indexNodeParent”. Semua node yang dibuat pada saat pembangunan sebuah tree disimpan dalam *array of struct* bernama “list_of_node”, sehingga untuk mengakses sebuah node digunakan indeks array dari “list_of_node”. Ilustrasi dari struktur data tersebut dapat dilihat pada Gambar 3.6, sedangkan penjelasan tiap member dari struct “node” terdapat dalam Tabel 3.2.



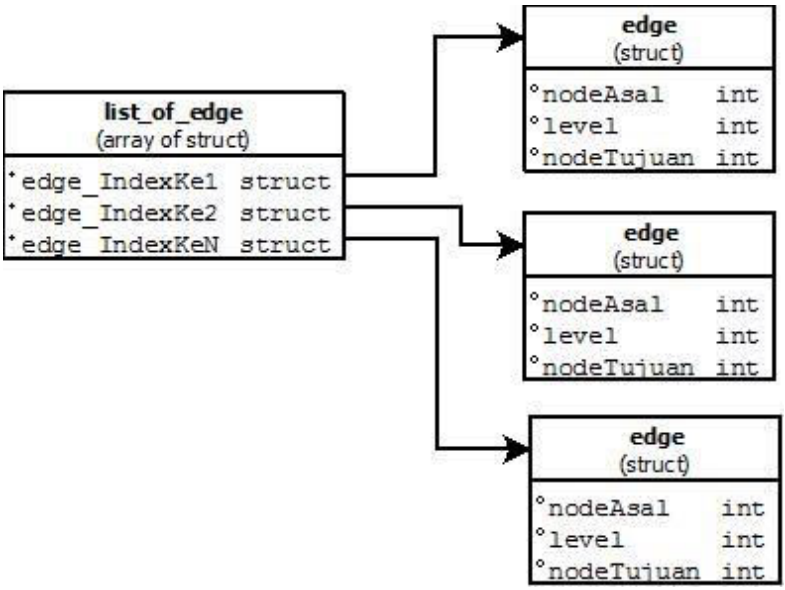
Gambar 3.6 Representasi Struktur Data Node

Tabel 3.2 Penjelasan Member Struct "node"

No	Member	Tipe	Penjelasan
1.	isLeaf	boolean	Bernilai 1 atau 0. Untuk menandai suatu *node* merupakan *leaf* atau bukan
2.	class	int	Menyimpan nilai kelas suatu *node*. Meskipun bukan *leaf* (isLeaf = 0), variabel ini akan tetap memiliki nilai yang ditentukan dari kelas yang sering muncul pada subset yang diklasifikasikan oleh node yang berkaitan. Hal ini akan berguna saat proses pruning.
3.	atribut	int	Menyimpan nilai atribut ke-i yang direpresentasikan oleh node.
4.	threshold	double	Menyimpan nilai threshold dari atribut ke-i. Nilai threshold ini digunakan untuk membagi set menjadi dua subset, yaitu <=threshold, dan >threshold.
5.	index	int	Menyimpan alamat indeks array dimana node yang berkaitan disimpan di dalam list_of_node
6.	pheromones	int	Menyimpan nilai feromon
7.	indexEdgeK	int	Merupakan indeks array dimana edge yang menuju ke subset "kurang dari sama dengan" disimpan dalam list_of_edge
8.	indexNodeParent	int	Merupakan indeks array dimana node parent dari node yang berkaitan disimpan dalam list_of_node
9.	indexEdgeL	int	Merupakan indeks array dimana edge yang menuju ke subset "lebih dari" disimpan dalam list_of_edge

3.4.2 Representasi Struktur Data Edge

Dalam tugas akhir ini sebuah edge direpresentasikan sebagai sebuah struct dengan tiga buah member yaitu “nodeAsal”, “level”, “nodeTujuan”. ”. Semua edge yang dibuat pada saat pembangunan sebuah tree disimpan dalam *array of struct* bernama “list_of_edge”, sehingga untuk mengakses sebuah edge digunakan indeks array dari “list_of_edge”. Ilustrasi dari struktur data tersebut dapat dilihat pada Gambar 3.7, sedangkan penjelasan tiap member dari struct “edge” terdapat dalam Tabel 3.3 (a dan b).



Gambar 3.7 Representasi Struktur Data Edge

Tabel 3.3 (a) Penjelasan Member Struct "edge"

No	Member	Tipe	Penjelasan
1.	nodeAsal	int	untuk menyimpan nilai index dalam list_of_node, yang merepresentasikan node mana

Tabel 3.3 (b) Penjelasan Member Struct "edge"

2.	level	int	untuk menyimpan nilai level dari edge. Nilai level ini menunjukkan di tingkat mana edge tersebut berada dalam suatu tree.
3.	nodeTujuan	int	untuk menyimpan nilai index dalam list_of_node, yang merepresentasikan node mana yang merupakan node tujuan dari edge tersebut

3.4.3 Representasi Struktur Data Feromon

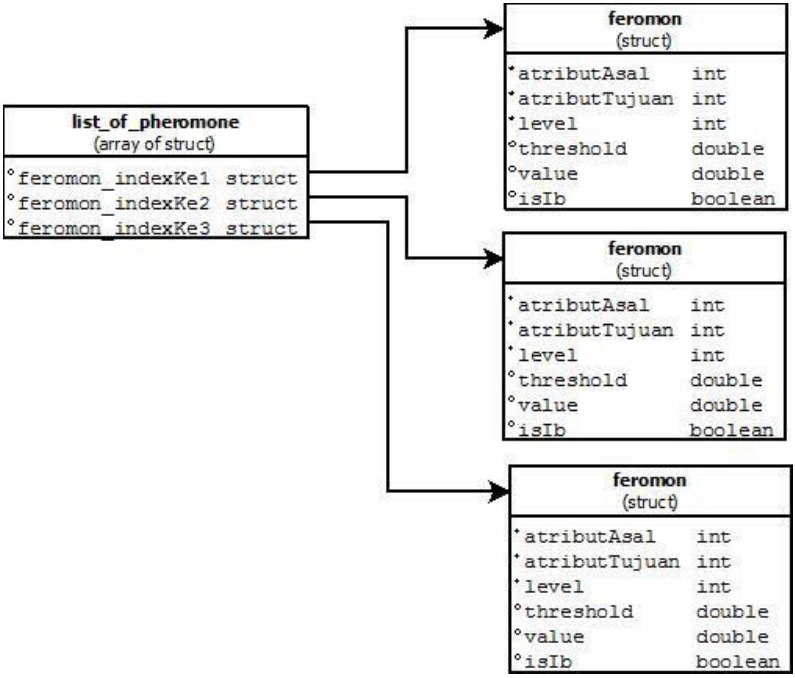
Dalam tugas akhir ini feromon-feromon pada tree disimpan dalam *array of struct* bernama list_of_pheromone. Tujuan dari penyimpanan feromon dalam sebuah array tersendiri adalah karena terdapat kemungkinan antara satu semut dengan satu semut yang lain membuat jalur yang sama, sehingga apabila jalur sudah pernah dibuat nilai feromon tinggal dipakai. Tetapi apabila jalur belum pernah dibuat maka akan dibuat data feromon baru dalam "list_of_pheromone". Masing-masing feromon merupakan sebuah struct dengan enam buah member yaitu "atributAsal", "atributTujuan", "level", "threshold", "value", "isIb". Struktur data tersebut dapat dilihat pada Gambar 3.8. Penjelasan tiap member terdapat pada Tabel 3.4 (a dan b).

Tabel 3.4 (a) Penjelasan Member struct "pheromone"

No	Member	Tipe	Penjelasan
1.	atributAsal	int	Menyimpan atribut asal (menunjukkan letak feromon dalam tree)
2.	atributTujuan	int	Menyimpan atribut tujuan (menunjukkan letak feromon dalam tree).

Tabel 3.4 (b) Penjelasan Member struct "pheromone"

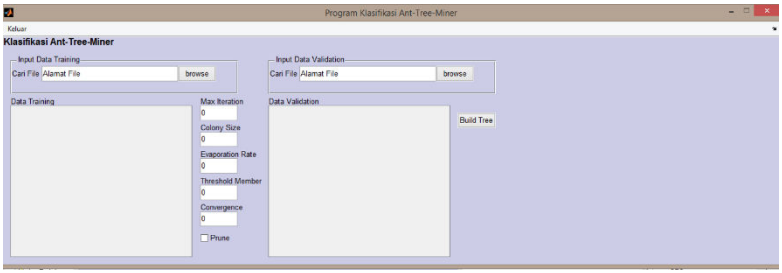
3.	level	int	untuk menyimpan di level mana dalam tree feromon ini berada
4.	threshold	double	Menyimpan nilai threshold (menunjukkan letak feromon dalam tree)
5.	value	double	Menyimpan nilai feromon
6.	isIb	boolean	Penanda apakah feromon tersebut merupakan *iteration best*, sehingga nanti akan berguna pada proses *update* feromon.



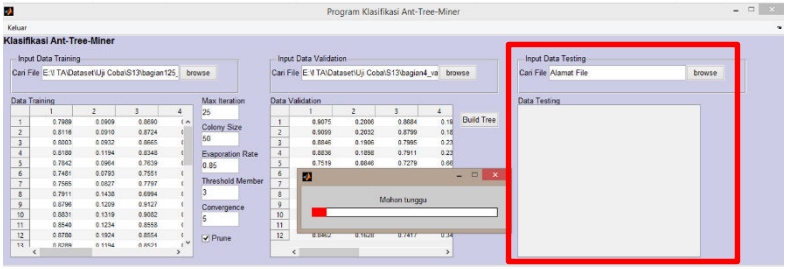
Gambar 3.8 Representasi Struktur Data Daftar Feromon

3.5 Perancangan Antarmuka Aplikasi

Gambar 3.9 menunjukkan rancangan antarmuka untuk perangkat lunak Klasifikasi Ikan Tuna Menggunakan Algoritma Ant-Tree-Miner. Penggunaan perangkat lunak ini tidak terbatas pada data ikan tuna, tetapi dapat juga untuk data lain dengan karakteristik semua aribut prediktornya bertipe kontinyu.



Gambar 3.9 Tampilan Awal Perangkat Lunak

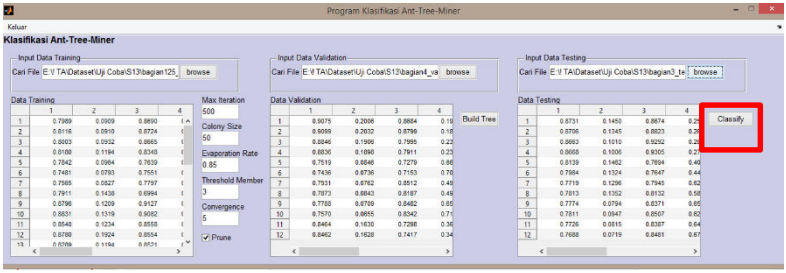


Gambar 3.10 Input Panel Data Testing

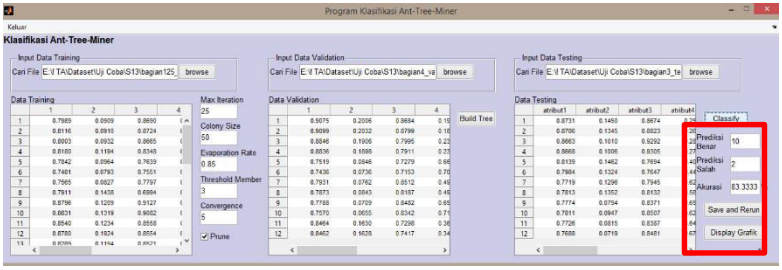
Pengguna akan memasukkan data training dan data validation dengan menekan tombol “browse” di input panel. Selain itu pengguna juga memasukkan lima buah parameter yaitu Max Iteration, Colony Size, Evaporation Rate, Threshold Member, dan Convergence pada textbox. Setelah menekan tombol “build tree”, input panel data testing akan muncul seperti ditunjukkan pada Gambar 3.10. Selain itu juga tampil kotak dialog yang menampilkan pesan “mohon tunggu”. Kotak pesan ini bertujuan

agar pengguna tidak memasukkan data testing terlebih dahulu sebelum proses pembangunan tree selesai, karena proses ini dapat memakan waktu lama bergantung pada parameter-parameter yang dimasukkan.

Setelah memasukkan data testing, tombol “classify” akan muncul. Tombol ini berfungsi untuk melakukan proses klasifikasi. Ketika tombol tersebut ditekan maka akan muncul teks yang menunjukkan jumlah klasifikasi benar, klasifikasi salah, dan persentase akurasi.



Gambar 3.11 Tombol "Classify"



Gambar 3.12 Kotak Akurasi dan Dua Tombol

Tombol “Save and Rerun” berfungsi untuk menyimpan hasil prediksi dan mengulangi proses dengan data input yang sama, mengingat hasil tree dibangun secara stokastik maka dengan input yang sama belum tentu akan menghasilkan hasil prediksi yang sama. Sedangkan tombol “Display Grafik” akan menampilkan grafik akurasi dari proses klasifikasi yang telah dilakukan, serta

akurasi rata-rata apabila proses klasifikasi dilakukan lebih dari satu kali.

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya. Sebelum penjelasan implementasi akan ditunjukkan terlebih dahulu lingkungan untuk melakukan implementasi.

4.1 Lingkungan Implementasi

Implementasi aplikasi klasifikasi ikan tuna menggunakan perangkat keras komputer dengan prosesor yang mempunyai spesifikasi Intel(R) Core(TM) i7 – 3610QM CPU @ 2.30GHz yang mempunyai memori sebesar 4.0 GB. Untuk sistem operasi yang digunakan adalah Windows 8.1 Pro 64-bit, dalam melakukan implementasi program menggunakan perangkat lunak Matlab R2013a.

4.2 Implementasi Algoritma Ant-Tree-Miner

Implementasi algoritma Ant-Tree-Miner dilakukan sesuai dengan diagram alir pada Gambar 2.6. Fungsi utama ini membutuhkan dua masukan data yaitu data training dan data validation. Sedangkan parameter yang dimasukkan adalah iterasi maksimum, jumlah semum, tingkat penguapan, threshold member, konvergensi maksimum, dan variabel penanda dilakukan pruning atau tidak. Kode sumber program utama dari algoritma Ant-Tree-Miner dapat dilihat pada Kode Sumber 4.1 (a dan b).

```
1.  function [treeGB m] = ATM(dataTraining,
    dataValidation, maxIteration, colonySize,
    evaporationRate, maxMember,
    maxConvergence, prune)
2.      P = [];
3.      treeGB = [];
```

```
4.      m = 0;
5.      convergence(1:maxConvergence) = 0;
6.      Qgb = -1;
7.      isConvergent = 0;
8.      while(m < maxIteration && isConvergent
        == 0)
9.          treeIB = [];
10.         Qib = -1;
11.         for n=1:colonySize
12.             [tree P] =
                createSingleTree(dataTraining, P,
                maxMember);
13.             prunedTree =
                pruning(dataValidation, tree);
14.             Q = computeQ(prunedTree,
                dataValidation);
15.             if (prune == 1)
16.                 tree = pruning(dataValidation,
                    tree);
17.             end;
18.             Q = computeQ(tree,
                dataValidation);
19.             if(Q > Qib || Qib == -1)
20.                 Qib = Q;
21.                 treeIB = tree;
22.             end;
23.         if(Qgb == convergence(mod(m,
            maxConvergence)+1))
24.             isConvergent = 1;
25.         end;
26.         convergence(mod(m,
            maxConvergence)+1) = Qgb;
27.         m = m+1;
28.     end
29. end
```

Kode Sumber 4.1 Implementasi Algoritma ATM

4.2.1 Pembangunan Tree

Pembangunan tree dilakukan oleh satu semut. Sehingga proses ini akan diulangi sebanyak parameter jumlah semut yang dimasukkan pada fungsi utama. Proses pembangunan tree dilakukan sesuai pada diagram alir padabab sebelumnya yaitu pada Gambar 3.3. Implementasi proses pembangunan tree dapat dilihat pada Kode Sumber 4.2.

```
1.  function [nodes, edges, indexNodeParent,
    list_of_pheromone]  =
    createTree(nodes, edges, examples,
    level, list_of_pheromone, atributAsal,
    maxMember)
2.
    %diskritisasi dan hitung informasi
3.  heuristi tiap atribut
4.  for i = 1:size(examples, 2)-1
            [gainRatio(i) thresholdValue(i)]
5.  = discrandFindGainRatio(examples, i);
6.
7.  %--------------------------------------
8.  %     ACO part
9.  %--------------------------------------
10.
11. %pheromoneReference berisi kumpulan
    index feromon dalam list_of_pheromone
12.         pheromoneReference(i) =
    findPheromon(atributAsal, i, level,
    thresholdValue(i),...
                    list_of_pheromone);
13. %jika feromon tidak ditemukan, buat
    feromon baru dalam list_of_pheromon
14.         if(pheromoneReference(i) == 0)
                indexP =
    size(list_of_pheromone,2)+1;
```

```
15.
      list_of_pheromone(indexP).atributAsal =
      atributAsal;
16.
      list_of_pheromone(indexP).atributTujuan
17.   = i;
18.
      list_of_pheromone(indexP).level = level;
19.
      list_of_pheromone(indexP).threshold =
20.   thresholdValue(i);
21.
22.   list_of_pheromone(indexP).value = 1;
23.
      list_of_pheromone(indexP).isIb = 0;
                  pheromoneReference(i) =
24.   indexP;
25.           end
26.       end;
27.
28.       indexClass = size(examples, 2);
29.
      %menghitung probability node i untuk
30.   dikunjungi
31.      for i=1:size(pheromoneReference, 2)
            pheromonUsed(i) =
      list_of_pheromone(pheromoneReference(i))
32.   .value;
33.       end
34.       pmulg = pheromonUsed.*gainRatio;
35.       probability = pmulg./sum(pmulg);
36.
37.   %membuat range kumulatif dari
38.   probability
39.       for i=1:size(probability, 2)
40.           cdf(i) = sum(probability(1:i));
41.       end


42.
```

```
%Buat bilangan random [0, 1],
selanjutnya disesuaikan pada range
kumulatif untuk menunjukkan atribut yang
dipilih semut
    random = rand(1);
    for i=1:size(cdf,2)
        if(random < cdf(i))
            atributPilihan = i;
            break;
        end
    end
%-------------------------------------
%     end of ACO part
%-------------------------------------

    %Buat parent node
    indexNodeParent = size(nodes,2)+1;
    nodes(indexNodeParent).isLeaf = 0;
    nodes(indexNodeParent).class =
mode(examples(:, indexClass));
    nodes(indexNodeParent).atribut =
atributPilihan;
    nodes(indexNodeParent).threshold =
thresholdValue(atributPilihan);
    nodes(indexNodeParent).index =
indexNodeParent;

% ACO, set nilai feromon pada node
    nodes(indexNodeParent).pheromones =
pheromoneReference(atributPilihan);

%buat subset dari examples sesuai
threshold. Terbentuk 2 subset yaitu
subset kurang dari sama dengan t dan
lebih dari t
    indexKurangDariSamaDengan =
examples(:,atributPilihan)<=thresholdVal
ue(atributPilihan);
```

```
68.     subsetKurangDariSamaDengan =
70. examples(indexKurangDariSamaDengan,:);
71.     subsetLebihDari =
    examples(~indexKurangDariSamaDengan,:);

72. %subset tree kurang dari sama dengan
    %-----------------------------------
73.     size_subsetKurangDariSamaDengan =
    size(subsetKurangDariSamaDengan,1);
74.
75.
76. %jika subset kosong buat leaf dengan
    label kelas mayoritas
77.     if(size_subsetKurangDariSamaDengan
    == 0 )
78.
79.
80.         indexNodeK = size(nodes,2)+1;
           nodes(indexNodeK).isLeaf = 1;
           nodes(indexNodeK).class =
    mode(examples(:, indexClass ));
           nodes(indexNodeK).atribut = 0;
82.        nodes(indexNodeK).threshold = 0;

83. nodes(indexNodeK).indexNodeParent =
    indexNodeParent;

84.        indexEdgeLeafNodeK =
    size(edges,2)+1;
85.
    edges(indexEdgeLeafNodeK).nodeAsal =
    indexNodeParent;
86.
    edges(indexEdgeLeafNodeK).nodeTujuan =
87. indexNodeK;
           edges(indexEdgeLeafNodeK).level
    = level+1;
88

89. nodes(indexNodeParent).indexEdgeK =
    indexEdgeLeafNodeK;
```

```
90.             nodes(indexNodeK).index =
    indexNodeK;

91.     %Jika subset mempunyai label kelas yang
    sama
        %Jika jumlah anggota subset kurang dari
92.     threshold
        %Buat leaf dengan kelas mayoritas

93.     elseif(size_subsetKurangDariSamaDengan
    ~= 0 && ...

94.     (histc(subsetKurangDariSamaDengan(:,
    indexClass),...
95.
    subsetKurangDariSamaDengan(1,indexClass)
96.     )...
97.
98.     ==size_subsetKurangDariSamaDengan ||...
99.
    size_subsetKurangDariSamaDengan <
    maxMember))
100.
                indexNodeK = size(nodes,2)+1;
                nodes(indexNodeK).isLeaf = 1;
101.            nodes(indexNodeK).class =
102.    mode(subsetKurangDariSamaDengan(:,indexC
    lass));
                nodes(indexNodeK).atribut = 0;
103.            nodes(indexNodeK).threshold = 0;

                indexEdgeLeafNodeK =
104.    size(edges,2)+1;

    edges(indexEdgeLeafNodeK).nodeAsal =
105.    indexNodeParent;

106.    edges(indexEdgeLeafNodeK).nodeTujuan =
    indexNodeK;
```

```
        edges(indexEdgeLeafNodeK).level
= level+1;
107.
        nodes(indexNodeParent).indexEdgeK =
108. indexEdgeLeafNodeK;
        nodes(indexNodeK).index =
indexNodeK;
109.
        nodes(indexNodeK).indexNodeParent =
indexNodeParent;
110.
111.   %buat subtree
112.       else
113.           indexEdgeK = size(edges,2)+1;
           edges(indexEdgeK).nodeAsal =
114. indexNodeParent;
           edges(indexEdgeK).level = level;
115.
           [nodes, edges,
116. edges(indexEdgeK).nodeTujuan,
117. list_of_pheromone] = ...
           createTree(nodes, edges,...
118.
        subsetKurangDariSamaDengan, level+1,
        list_of_pheromone,...
119.
        nodes(indexNodeParent).atribut,
120. maxMember);
121.
122.
        nodes(indexNodeParent).indexEdgeK =
123. indexEdgeK;
124.
        nodes(edges(indexEdgeK).nodeTujuan).inde
125. xNodeParent = indexNodeParent;
126.
127.       end
128.
```

```
129.
        %subset tree lebih dari
130.    %-----------------------------------
            size_subsetLebihDari =
131.    size(subsetLebihDari,1);
132.
        %jika subset kosong buat leaf dengan
133.    label kelas mayoritas
134.        if(size_subsetLebihDari == 0 )
135.            indexNodeL = size(nodes,2)+1;
136.            nodes(indexNodeL).isLeaf = 1;
137.
138.            nodes(indexNodeL).class =
        mode(examples(:, indexClass ) );
139.            nodes(indexNodeL).atribut = 0;
140.            nodes(indexNodeL).threshold = 0;
141.
        nodes(indexNodeL).indexNodeParent =
142.     indexNodeParent;
143.
                indexEdgeLeafNodeL =
144.    size(edges,2)+1;
145.
        edges(indexEdgeLeafNodeL).nodeAsal =
146.     indexNodeParent;
147.
        edges(indexEdgeLeafNodeL).nodeTujuan =
148.     indexNodeL;
                edges(indexEdgeLeafNodeL).level
149     = level+1;
150.
        nodes(indexNodeParent).indexEdgeL =
151.     indexEdgeLeafNodeL;
                nodes(indexNodeL).index =
152.    indexNodeL;
153.
154.
```

```
%Jika subset mempunyai label kelas yang sama
%Jika jumlah anggota subset kurang dari threshold
%Buat leaf dengan kelas mayoritas
    elseif(size_subsetLebihDari ~= 0 && ...
            (histc(subsetLebihDari(:,indexClass),subsetLebihDari(1,indexClass))...
            ==size_subsetLebihDari ||...
size_subsetKurangDariSamaDengan < maxMember))

        indexNodeL = size(nodes,2)+1;
        nodes(indexNodeL).isLeaf = 1;
        nodes(indexNodeL).class = mode(subsetLebihDari(:,indexClass));
        nodes(indexNodeL).atribut = 0;
        nodes(indexNodeL).threshold = 0;

nodes(indexNodeL).indexNodeParent = indexNodeParent;

        indexEdgeLeafNodeL = size(edges,2)+1;

edges(indexEdgeLeafNodeL).nodeAsal = indexNodeParent;

edges(indexEdgeLeafNodeL).nodeTujuan = indexNodeL;
        edges(indexEdgeLeafNodeL).level = level+1;


nodes(indexNodeParent).indexEdgeL = indexEdgeLeafNodeL;
```

```
183.          nodes(indexNodeL).index =
184. indexNodeL;
185.
186. %buat subtree
         else
187.          indexEdgeL = size(edges,2)+1;
188.          edges(indexEdgeL).nodeAsal =
189. indexNodeParent;
              edges(indexEdgeL).level = level;

190.
191.          [nodes, edges,
192. edges(indexEdgeL).nodeTujuan,
193  list_of_pheromone] = ...
194.                createTree(nodes, edges,...
                                subsetLebihDari,
     level+1, list_of_pheromone,...

195.
196. nodes(indexNodeParent).atribut,
     maxMember );


197.
198.        nodes(indexNodeParent).indexEdgeL
     = indexEdgeL;
199.
200. nodes(edges(indexEdgeL).nodeTujuan).inde
     xNodeParent = indexNodeParent;
         end
     end
```

Kode Sumber 4.2 Implementasi Pembangunan Tree

4.2.2 Implementasi Proses Pruning

Proses pruning mengikuti alur yang sudah dijelaskan pada bab sebelumnya, tepatnya pada gambar 3.4. Implementasi proses pruning dapat dilihat pada Kode Sumber 4.3.

```
1.  function [prunedTree] =
    pruning(dataValidation, tree)
2.      for i=1:size(dataValidation, 1)
            [class(i), indexLeaf(i)] =
    classifyPrune(tree.list_of_node(tree.roo
3.  t),...

    dataValidation(i, :),...

4.  tree.list_of_node, tree.list_of_edge);

5.      end

6.      list_of_node = tree.list_of_node;
        list_of_edge = tree.list_of_edge;
7.
8.
    %     mengisi error table
        errorTable(:, 1) =
    unique(indexLeaf);
9.
10.     errorTable(:, 2) = 0;
        for i=1:size(dataValidation, 1)
11.         index = find(errorTable(:,
    1)==indexLeaf(i));
12.
13.         errorTable(index, 2) =
    errorTable(index, 2)+...

14.
    (class(i)~=dataValidation(i,9));
        end
        errorTable(:, 3) =
    histc(indexLeaf,unique(indexLeaf));
15.
16.
    %     pruning
        masihBisaPrune = 1;
17.     while(masihBisaPrune == 1 )
18.
19.        errorTableBaru = [];
           for i=1:size(errorTable, 1)
20.
21.
```

```
22.             indexParent =
list_of_node(errorTable(i,
1)).indexNodeParent;

23.             indexedgeK =
list_of_node(indexParent).indexEdgeK;
24.             indexChildK =
list_of_edge(indexedgeK).nodeTujuan;

25.             indexedgeL =
list_of_node(indexParent).indexEdgeL;
26.             indexChildL =
list_of_edge(indexedgeL).nodeTujuan;


27.  if(list_of_node(indexChildK).isLeaf==1
     &&...

     list_of_node(indexChildL).isLeaf==1)
                    %jika kedua child
     dilewati

28.  if(~isempty(find(errorTable(:,
     1)==indexChildK)) &&...

     ~isempty(find(errorTable(:,
     1)==indexChildL)))
                         isPruned =
29.  checkPrune(indexParent, indexChildK,...

     indexChildL, list_of_node, errorTable);

                         %cek perlu diprune
     atau tidak berdasar error rate
30.                      if(isPruned)
                             [tree
     errorTableBaru class indexLeaf] =...
```

```
31.  parentToLeaf(tree, indexParent,
     indexChildK,...

     indexChildL, class, indexLeaf);
32.                      end

                    %leaf yang dilewati
     hanya yg kiri atau yg kanan aja

     elseif(~isempty(find(errorTable(:,
33.  1)==indexChildK)) ||...

     ~isempty(find(errorTable(:,
     1)==indexChildL)))
34.                      [tree errorTableBaru
     class indexLeaf] =...

     parentToLeaf(tree, indexParent,
     indexChildK,...

     indexChildL, class, indexLeaf);
35.                  end
36.              end
37.          end


38.          if(isempty(errorTableBaru))
39.              masihBisaPrune = 0;
40.          else
41.              errorTable = errorTableBaru;
             end;


42.      end
43.      prunedTree = tree;
44.  end
```

Kode Sumber 4.3 Implementasi Proses Pruning

BAB V
UJI COBA DAN EVALUASI

Pada bab ini akan dijelaskan uji coba yang dilakukan pada aplikasi yang telah dikerjakan serta analisa dari uji coba yang telah dilakukan. Pembahasan pengujian meliputi lingkungan uji coba, Pengujian algoritma dan aplikasi serta evaluasi uji coba.

5.1 Lingkungan Uji Coba

Dalam melakukan proses uji coba dan evaluasi dari Tugas Akhir ini menggunakan perangkat keras komputer dengan prosesor yang mempunyai spesifikasi Intel(R) Core(TM) i7 – 3610QM CPU @ 2.30GHz yang menggunakan memori sebesar 4.0 GB. Untuk sistem operasi yang digunakan adalah Windows 8.1 Pro 64-bit. Dalam melakukan implementasi program digunakan perangkat lunak Matlab R2013a.

5.2 Data Pengujian

Data masukan yang digunakan pada uji coba Tugas Akhir ini adalah data hasil ekstraksi fitur citra ikan tuna yang berbentuk numerik. Data masukan direpesentasikan kedalam file dengan format *Comma Separated Values* (CSV). Selain berisi nilai dari delapan atribut seperti dijelaskan pada bagian 3.3.1, pada kolom terakhir ditambahkan satu kolom yang berisi kelas sebenarnya dari tiap *record*. Nilai ini berfungsi untuk menghitung akurasi prediksi kelas.

Data masukan berjumlah sebanyak enam puluh data dengan tiga kelas yang berbeda masing-masing berjumlah dua puluh data.

5.3 Skenario Uji Coba

Skenario uji coba yang dilakukan dengan menggunakan data training sebanyak 60%, data validation sebanyak 20% dan data testing sebanyak 20% dari keseluruhan data. Skenario Uji Coba dilakukan untuk mengetahui dampak proses pruning, penentuan parameter jumlah koloni semut, dan penentuan parameter konvergensi, serta pengaruh jumlah iterasi pada performa algoritma ATM. Skenario uji coba dapat dilihat pada Tabel 5.1. Mengingat sifat algoritma yang melibatkan bilangan random, maka untuk setiap skenario akan diulang sebanyak lima belas kali, lalu akan dihitung rata-rata akurasi lima belas kali percobaan tersebut.

Tabel 5.1 Skenario Uji Coba

Skenario	Parameter Uji Coba	Nilai Uji
1	Pruning	Prune, Not Prune
2	Koloni Semut	50, 100, 200, 250, 300
3	Parameter Konvergensi	5, 10
4	Iterasi Maksimum	5, 10, 15, 20, 25, 30

5.4 Uji Coba Skenario 1

Skenario 1 menguji pengaruh proses pruning terhadap performa algoritma ATM. Dalam skenario ini parameter lain ditentukan sebagaimana dalam Tabel 5.2 (a dan b).

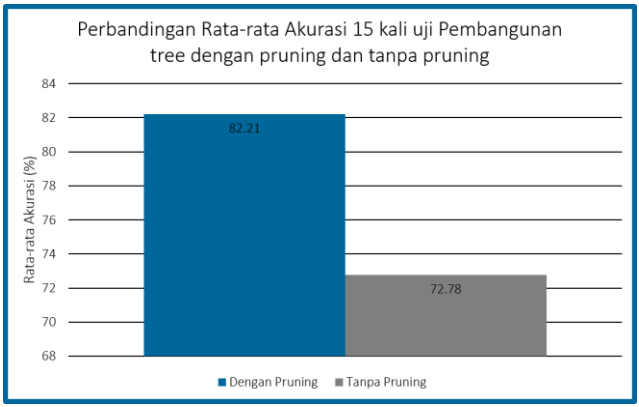
Gambar 5.1 merupakan hasil akurasi uji skenario 1. Grafik biru merupakan pembangunan tree dengan pruning, menghasilkan rata-rata akurasi 82,22%. Grafik abu-abu merupakan pembangunan tree tanpa pruning yang menghasilkan rata-rata akurasi 72,78%.

Tabel 5.2 (a) Parameter Lain dalam Uji Coba Skenario 1

Iterasi maksimum	500

Tabel 5.2(b) Parameter Lain dalam Uji Coba Skenario 1

Jumlah semut	200
Koefisien Evaporasi	0.85
Threshold member subset	3
Parameter konvergensi	5
Perulangan uji	15



Gambar 5.1 Hasil Uji Skenario 1

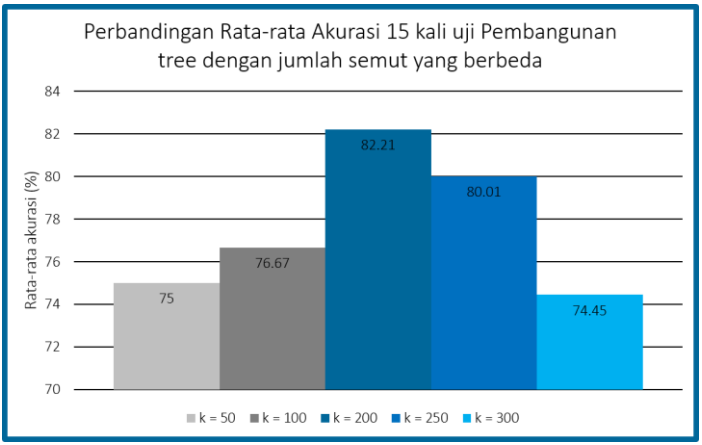
5.5 Uji Coba Skenario 2

Skenario 2 menguji pengaruh parameter jumlah koloni semut terhadap performa algoritma ATM. Dalam skenario ini parameter lain ditentukan sebagaimana dalam Tabel 5.3.

Tabel 5.3 Parameter Lain dalam Uji Coba Skenario 2

Iterasi maksimum	500
Koefisien Evaporasi	0.85
Threshold member subset	3
Parameter konvergensi	5
Perulangan uji	15
Pruning	ya

Pemilihan nilai koloni semut yang diuji coba berdasarkan pada penelitian [2]. Pada penelitian tersebut jumlah semut yang diuji coba adalah 50, 100, dan 200. Tetapi pada penelitian ini karena hasil akurasi masih membaik pada saat jumlah semut 200, maka diuji coba untuk jumlah semut yang lebih besar, yaitu 250 dan 300.



Gambar 5.2 Hasil Uji Skenario 2

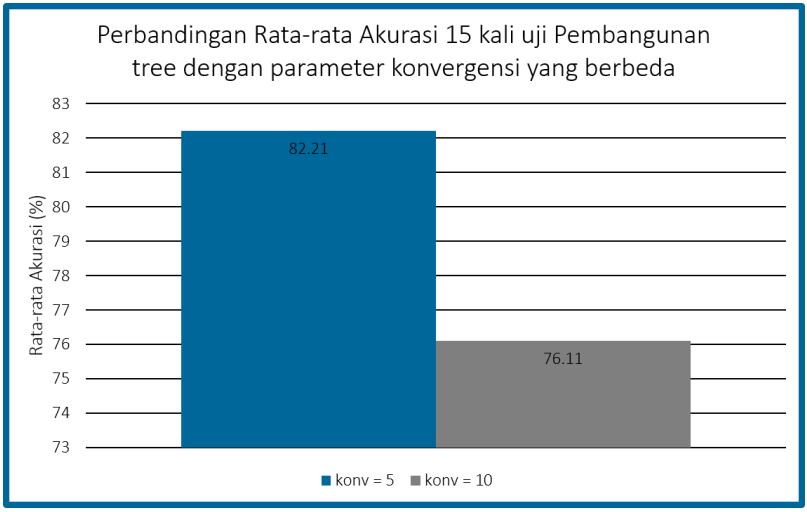
Gambar 5.2 merupakan hasil akurasi uji skenario 2. Grafik berwarna abu-abu muda merupakan hasil akurasi dengan k = 50, grafik berwarna abu-abu tua merupakan hasil rata-rata akurasi dengan k = 100, grafik berwarna biru tua merupakan hasil rata-rata akurasi dengan k = 200, grafik berwarna biru cerah merupakan hasil rata-rata akurasi dengan k = 250, dan terakhir grafik berwarna biru muda merupakan hasil rata-rata akurasi dengan k = 300. Hasil yang didapatkan dari lima uji coba tersebut berturut-turut adalah 75%; 76,67%; 82,21%; 80,01%; 74,45%. Dari hasil pengujian tersebut dapat diambil kesimpulan bahwa semut yang semakin banyak tidak serta-merta performa meningkat. Dibutuhkan jumlah semut yang pas untuk bisa mendapatkan performa terbaik algoritma.

5.5 Uji Kinerja Skenario 3

Skenario 3 menguji pengaruh parameter konvergensi terhadap performa algoritma ATM. Pemilihan nilai parameter konvergensi ditetapkan yaitu 5 dan 10. Dalam skenario ini parameter lain ditentukan sebagaimana dalam Tabel 5.4.

Tabel 5.4 Parameter Lain dalam Uji Coba Skenario 3

Iterasi maksimum	500
Jumlah semut	200
Koefisien Evaporasi	0.85
Threshold member subset	3
Parameter konvergensi	10
Perulangan uji	15



Gambar 5.3 Hasil Uji Skenario 3 untuk Nilai Konvergensi = 10

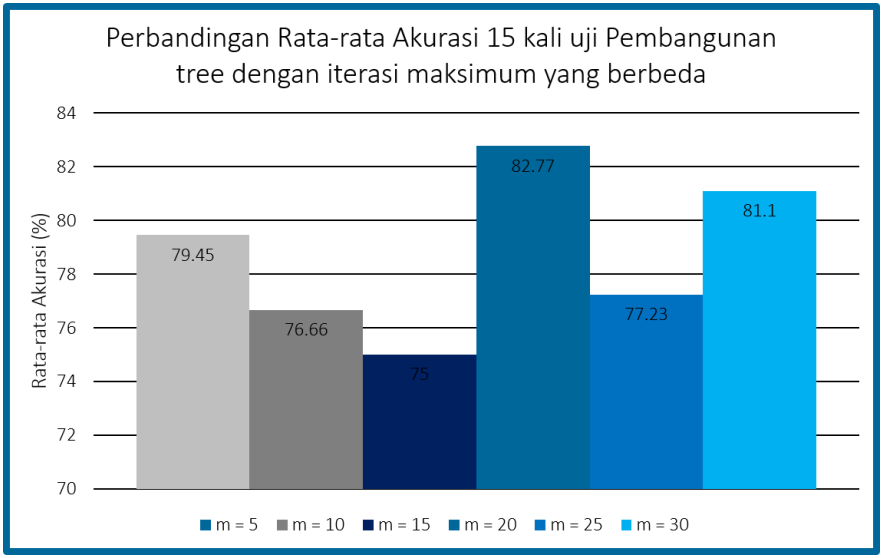
Gambar 5.3 merupakan hasil akurasi uji skenario 3. Akurasi rata-rata yang didapatkan adalah 82,21% dan 76,11% berturut-turut untuk parameter konvergensi 5 dan 10.

5.6 Uji Kinerja Skenario 4

Skenario 4 menguji pengaruh iterasi maksimum terhadap performa algoritma ATM. Dalam skenario ini parameter lain ditentukan sebagaimana dalam Tabel 5.5 (a dan b). Pada saat pengujian ini fungsi cek konvergensi dinonaktifkan supaya iterasi tidak berhenti sebelum iterasi yang diinginkan. Nilai-nilai iterasi maksimum yang diuji coba ini dipilih karena dengan iterasi 500 pun ternyata algoritma sudah konvergen pada iterasi yang kecil.

Tabel 5.5 Parameter Lain dalam Uji Skenario 4

Jumlah semut	200
Koefisien Evaporasi	0.85
Threshold member subset	3
Parameter konvergensi	0
Pruning	ya
Perulangan uji	15



Gambar 5.4 Hasil Uji Skenario 4

Gambar 5.4 merupakan hasil akurasi uji skenario 4. Berturut-turut dari kiri ke kanan adalah hasil rata-rata akurasi untuk iterasi maksimum m = 5, m = 10, m = 15, m = 20, m = 25, m = 30.

5.7 Evaluasi Hasil

Performa terbaik algoritma Ant-Tree-Miner di masing-masing skenario uji coba dapat dilihat pada Tabel 5.6. Rata-rata akurasi terbaik untuk skenario pertama, kedua, dan ketiga adalah 82,22% sedangkan untuk akurasi keempat adalah 82,77%.

Tabel 5.6 Ringkasan Hasil Uji Coba

Skenario	Parameter	Nilai	Rata-rata akurasi
Skenario 1	Pruning	Ya	82,22%
Skenario 2	Jumlah semut	200	82,22%
Skenario 3	Parameter konvergensi	5	82,22%
Skenario 4	Iterasi maksimum	20	82.77%

(Halaman ini sengaja dikosongkan)

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan dari hasil uji coba yang telah dilakukan sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, juga terdapat saran yang ditujukan untuk pengembangan perangkat lunak lebih lanjut.

6.1 Kesimpulan

Dari hasil uji coba yang telah dilakukan terhadap implementasi algoritma Ant-Tree-Miner untuk klasifikasi data ikan tuna, dapat diambil kesimpulan sebagai berikut:

1. Performa algoritma tidak menentu akibat terlibatnya bilangan random.
2. Pembangunan dengan proses pruning memberikan rata-rata akurasi lebih baik dibandingkan dengan pembangunan tanpa pruning
3. Jumlah koloni semut sebanyak 200 memberikan rata-rata akurasi paling baik dibandingkan dengan jumlah koloni lain yang diuji coba.
4. Jumlah iterasi maksimum m = 20 memberikan rata-rata akurasi paling baik dibandingkan dengan iterasi maksimum lain yang diuji coba.

6.2 Saran

Saran yang diberikan untuk pengembangan tugas akhir ini adalah karena data yang semua atributnya bersifat kontinyu, maka proses diskritisasi akan memegang peranan penting. Oleh sebab itu perlu dicoba melakukan modifikasi pada proses diskritisasi. Selain itu, hasil percobaan menunjukkan bahwa proses pruning mempengaruhi performa algoritma sehingga juga perlu dicoba melakukan modifikasi pada proses pruning yang dilakukan.

(Halaman ini sengaja dikosongkan)

DAFTAR PUSTAKA

[1] M. A. Kalbuadi, Penerapan Algoritma Decision Tree Pada Klasifikasi Citra Ikan Tuna. Studi Kasus: PT. Aneka Tuna Indonesia, Surabaya: Jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember Surabaya, 2014.

[2] F. E. Otero, "Inducing Decision Trees with an Ant Colony Optimization Algorithm," *Elsevier, Applied Soft Computing 12,* pp. 3615-3626, 2012.

[3] P. Alvarez, "Decision Tree Pruning based on Confidence Intervals (as in C4.5)," 15 Desember 2015. [Online]. Available: http://www.cs.bc.edu/~alvarez/ML/statPruning.html.

[4] P. W. Budi Santosa, Metoda Metaheuristik Konsep dan Implementasi, Surabaya: Penerbit Guna Widya, 2011.

(Halaman ini sengaja dikosongkan)

BIODATA PENULIS

Reinaldy Jalu Nusantara, Lahir di Surabaya, 8 Juni 1992. Penulis menempuh pendidikan mulai dari SDN Dr. Soetomo V Surabaya (lulus 2004), SMPN 1 Surabaya (lulus 2007), SMAN 5 Surabaya (lulus 2010) dan S1 Teknik Informatika ITS (2010-2015).

Selama masa kuliah, penulis aktif dalam beberapa kegiatan yang ada di lingkungan kampus ITS di organisasi tingkat jurusan, yaitu Himpunan Mahasiswa Teknik Computer-Informatika (HMTC) di Departemen Hubungan Luar Negeri, dan organisasi tingkat Fakultas, yaitu Badan Eksekutif Mahasiswa Fakultas Teknologi Informasi di Departemen Hubungan Luar. Penulis selama kuliah S1 mengambil bidang minat Dasar dan Terapan Komputasi (DTK). Penulis dapat dihubungi melalui surel: jalunusantara@gmail.com.